University of Zurich^UZH

s.e.a.l. software evolution & architecture lab

ICET LAB

CHALMERS

UNIVERSITY OF GOTHENBURG

https://www.flickr.com/photos/wetworkphotography/7485255952

Software Microbenchmarking in the Cloud. How Bad is it Really?

Christoph Laaber, Joel Scheuner, Philipp Leitner

laaber@ifi.uzh.ch

@ChristophLaaber

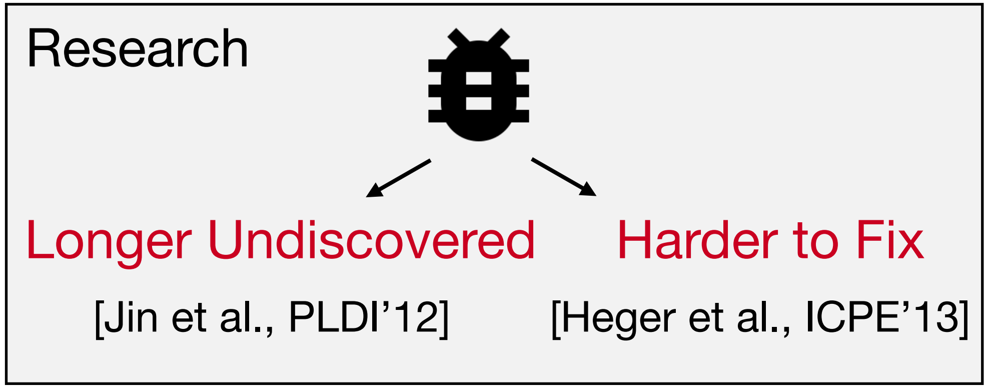
http://t.uzh.ch/T4

published in Empirical Software Engineering 24(4)

Journal-First

34th IEEE/ACM International Conference on Automated Software Engineering (ASE' 19), November 11 – 15, 2019, San Diego, CA, USA

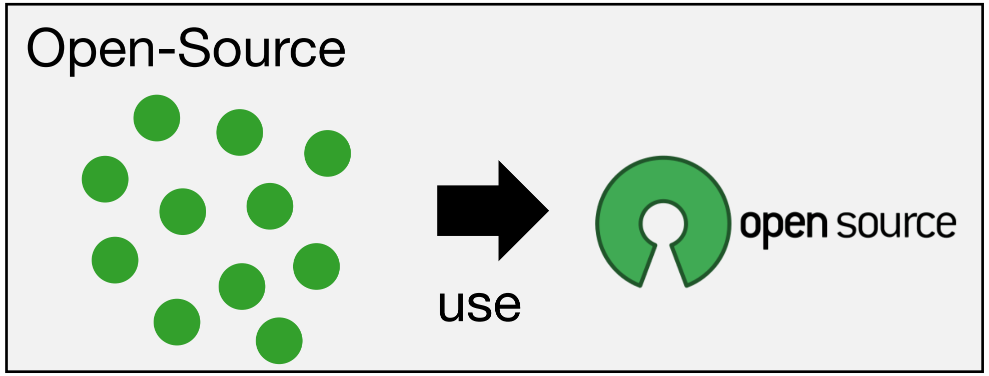
Why Software Performance Matters!



Research

Longer Undiscovered [Jin et al., PLDI’12]

Harder to Fix [Heger et al., ICPE’13]



Open-Source

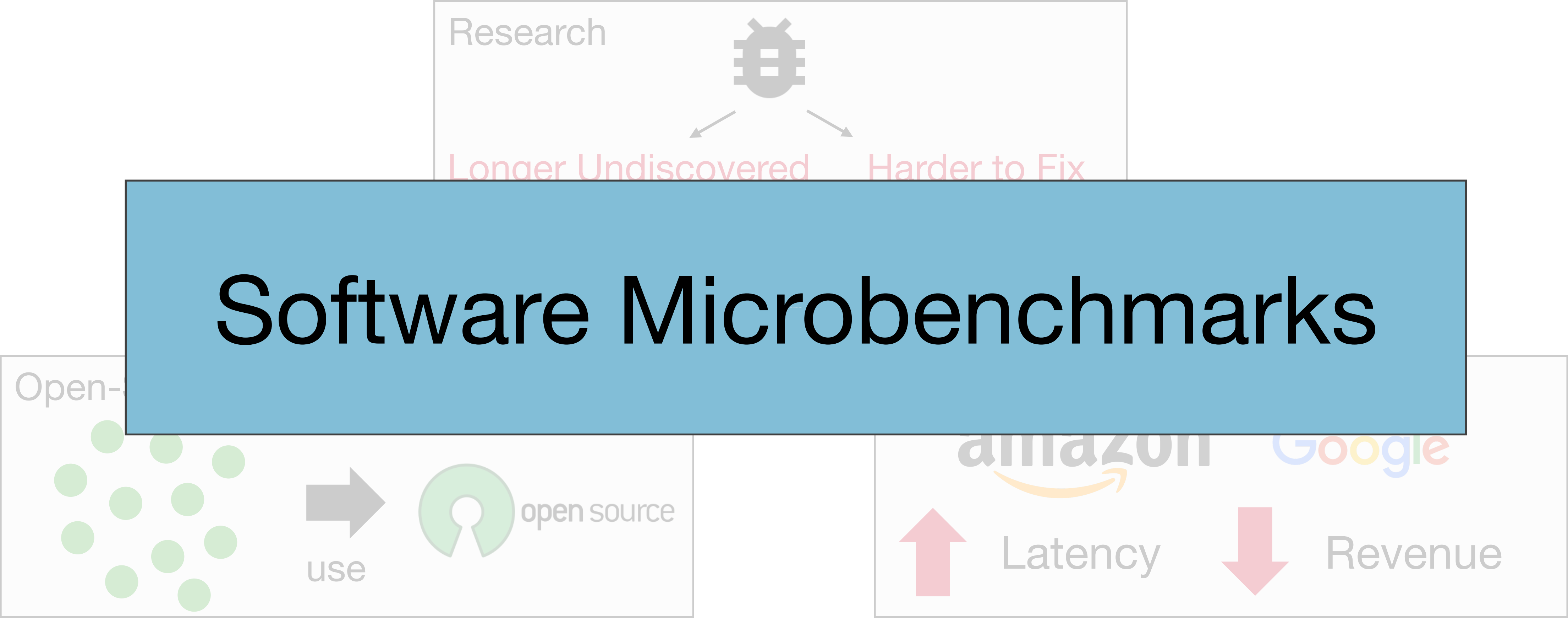
use

Industry

amazon
Google

↑ Latency ↓ Revenue

One Potential Solution



Software Microbenchmarks

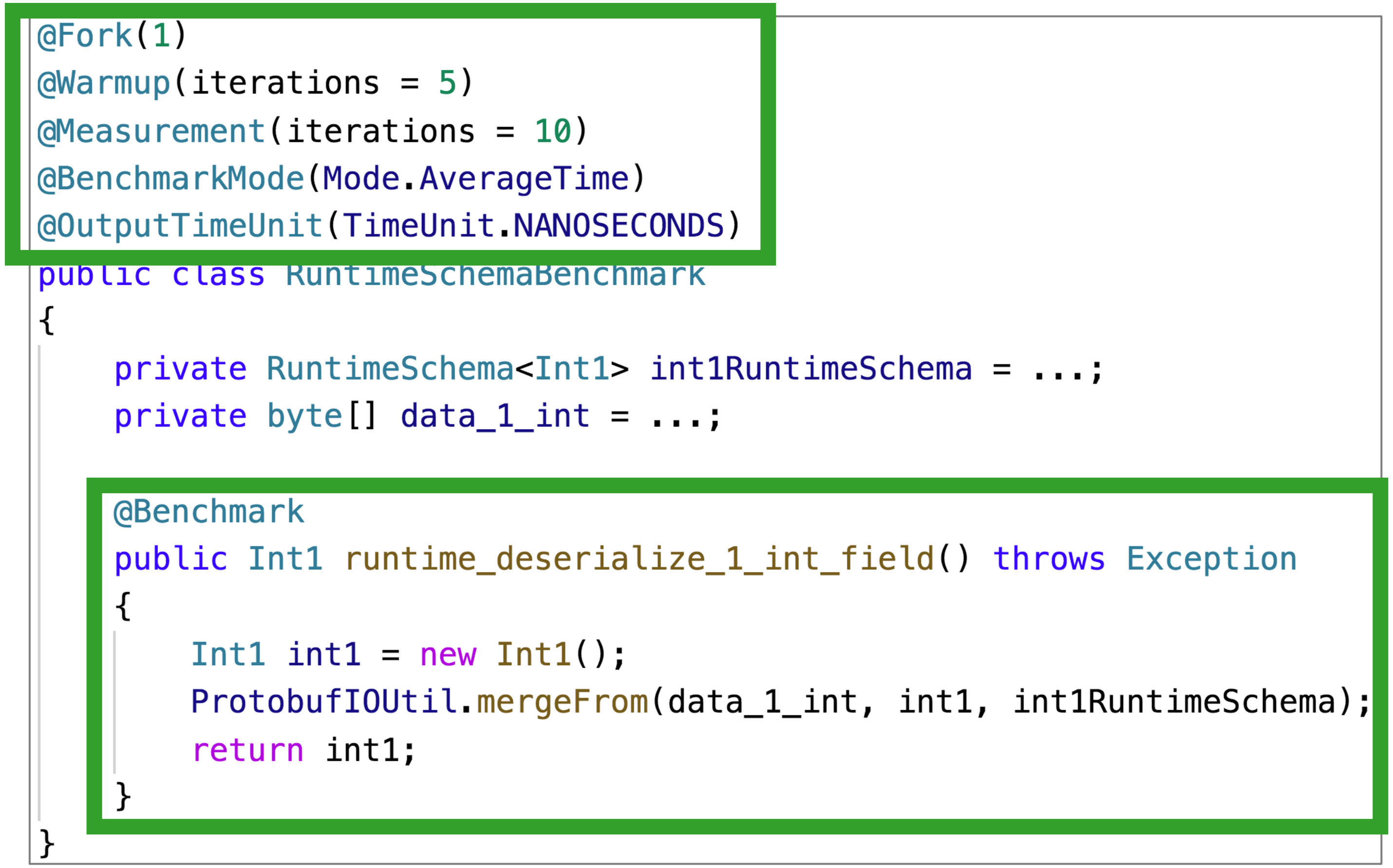
What are Software Microbenchmarks?

Benchmark

Performance Test

Unit test equivalent

Granularity: statement/method



```java
@Fork(1)
@Warmup(iterations = 5)
@Measurement(iterations = 10)
@BenchmarkMode(Mode.AverageTime)
@OutputTimeUnit(TimeUnit.NANOSECONDS)
public class RuntimeSchemaBenchmark
{
    private RuntimeSchema<Int1> int1RuntimeSchema = ...;
    private byte[] data_1_int = ...;

    @Benchmark
    public Int1 runtime_deserialize_1_int_field() throws Exception
    {
        Int1 int1 = new Int1();
        ProtobufIOUtil.mergeFrom(data_1_int, int1, int1RuntimeSchema);
        return int1;
    }
}
```

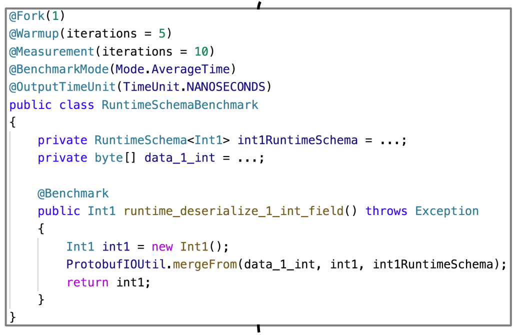
Execution Configuration

Implementation

What are Software Microbenchmarks?

Benchmark

Unit-level Performance Test



```
@Fork(1)
@Warmup(iterations = 5)
@Measurement(iterations = 10)
@BenchmarkMode(Mode.AverageTime)
@OutputTimeUnit(TimeUnit.NANOSECONDS)
public class RuntimeSchemaBenchmark
{
    private RuntimeSchema<Int1> int1RuntimeSchema = ...;
    private byte[] data_1_int = ...;

    @Benchmark
    public Int1 runtime_deserialize_1_int_field() throws Exception
    {
        Int1 int1 = new Int1();
        ProtobufIOUtil.mergeFrom(data_1_int, int1, int1RuntimeSchema);
        return int1;
    }
}
```

Statements/Methods

Execution

Iterations

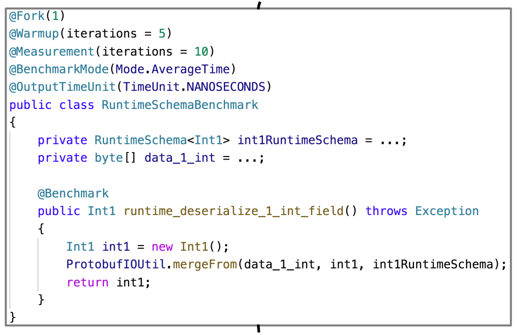
Machines

Trials

What are Software Microbenchmarks?

Benchmark

Unit-level Performance Test



```
@Fork(1)
@Warmup(iterations = 5)
@Measurement(iterations = 10)
@BenchmarkMode(Mode.AverageTime)
@OutputTimeUnit(TimeUnit.NANOSECONDS)
public class RuntimeSchemaBenchmark
{
    private RuntimeSchema<Int1> int1RuntimeSchema = ...;
    private byte[] data_1_int = ...;

    @Benchmark
    public Int1 runtime_deserialize_1_int_field() throws Exception
    {
        Int1 int1 = new Int1();
        ProtobufIOUtil.mergeFrom(data_1_int, int1, int1RuntimeSchema);
        return int1;
    }
}
```

Statements/Methods

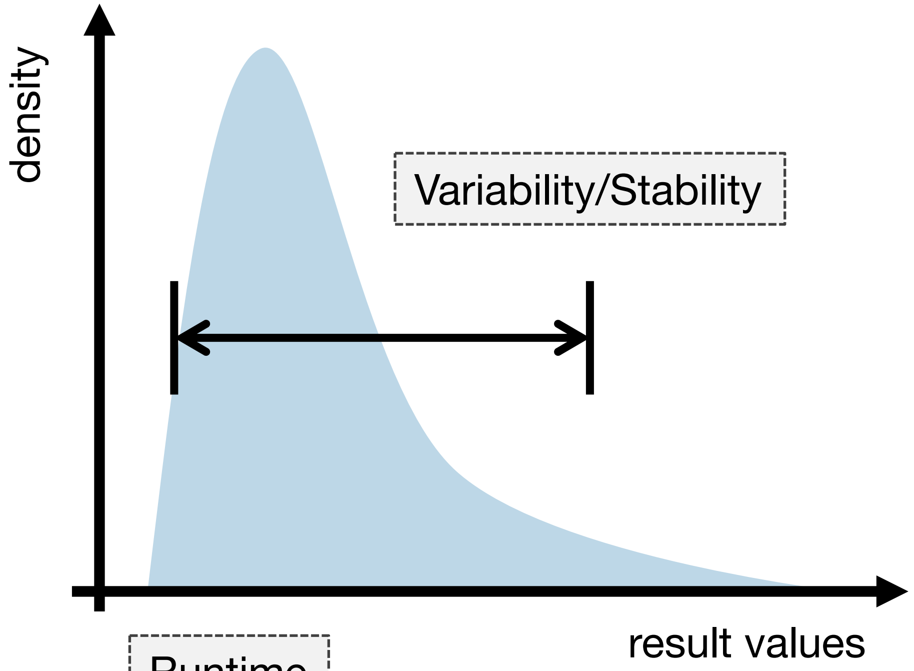
Execution

Iterations

Machines

Trials

Results



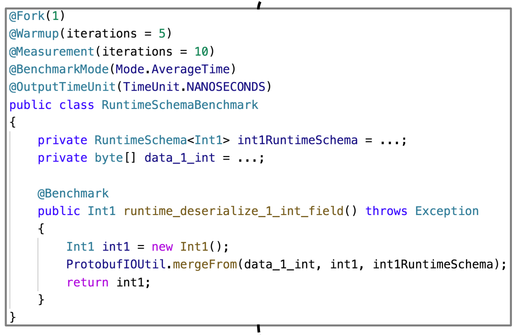
Runtime

Throughput

What are Software Microbenchmarks?

Benchmark

Unit-level Performance Test



```
@Fork(1)
@Warmup(iterations = 5)
@Measurement(iterations = 10)
@BenchmarkMode(Mode.AverageTime)
@OutputTimeUnit(TimeUnit.NANOSECONDS)
public class RuntimeSchemaBenchmark
{
    private RuntimeSchema<Int1> int1RuntimeSchema = ...;
    private byte[] data_1_int = ...;

    @Benchmark
    public Int1 runtime_deserialize_1_int_field() throws Exception
    {
        Int1 int1 = new Int1();
        ProtobufIOUtil.mergeFrom(data_1_int, int1, int1RuntimeSchema);
        return int1;
    }
}
```

Statements/Methods

Execution

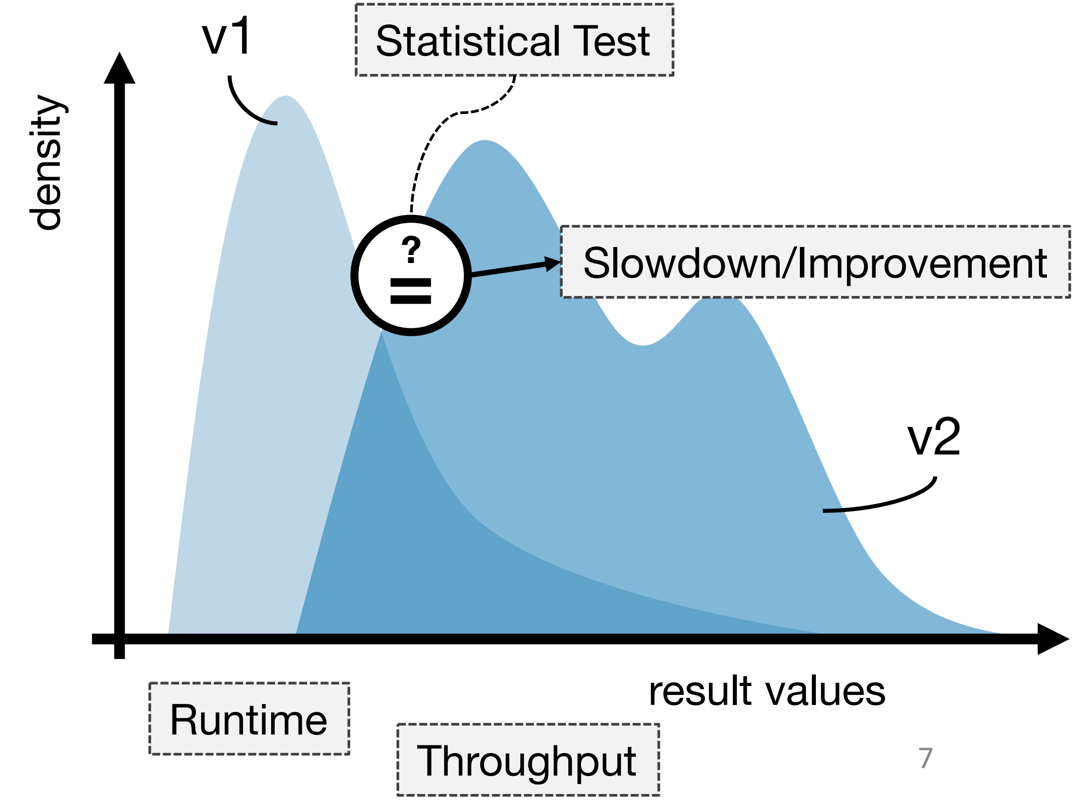
Iterations

Machines

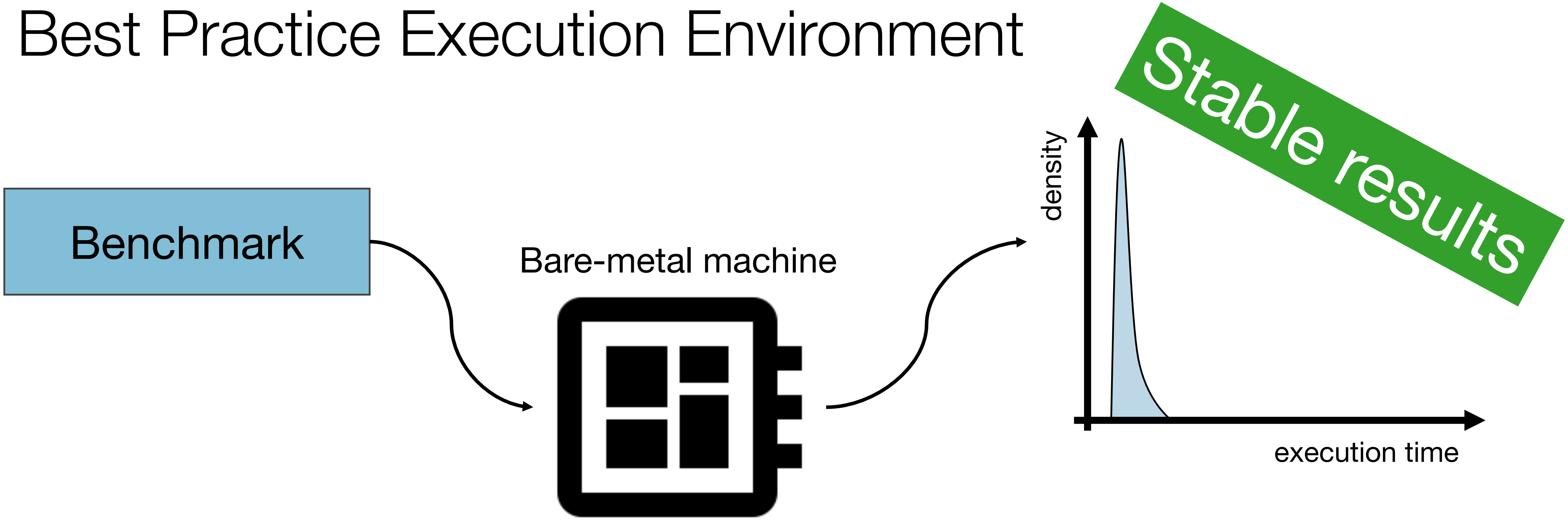
Trials

Results

Comparison



Best Practice Execution Environment



Benchmark

Bare-metal machine

Stable results

No virtualization

Single tenant

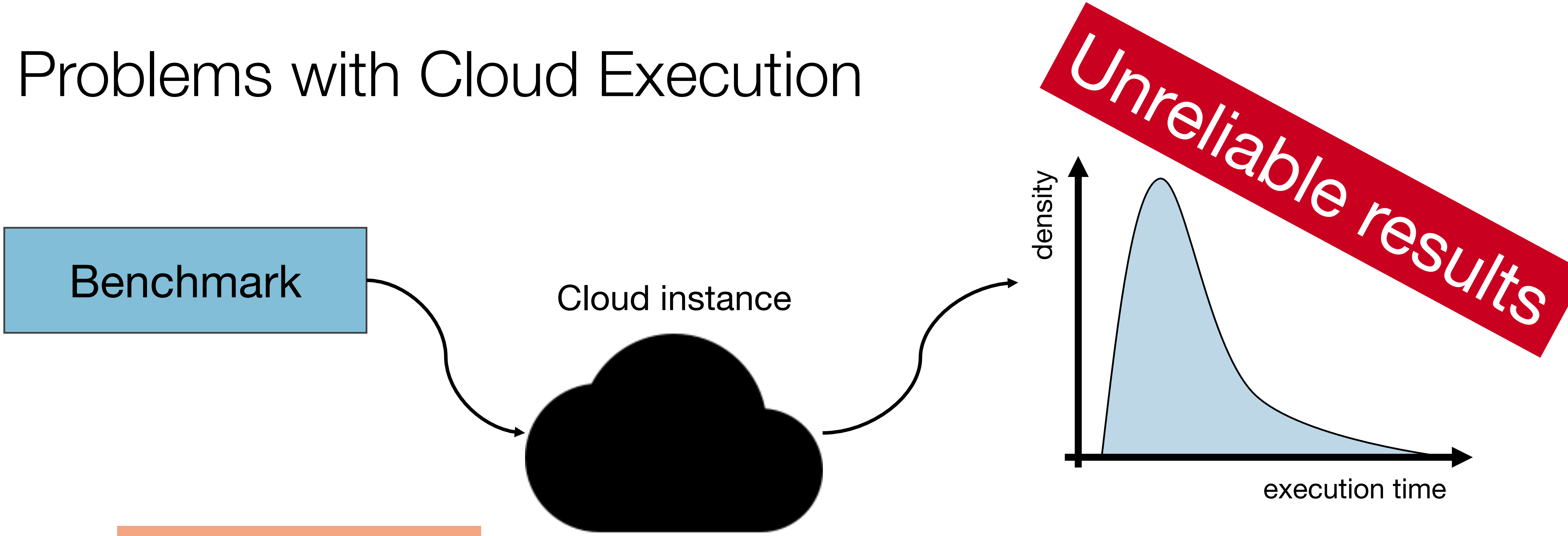
No hardware/software optimizations

No background processes/services

Why Execute Benchmarks in the Cloud then?

- Unavailability of / no training for bare-metal machines
- Long benchmarking run times
- Little set-up and maintenance effort
- Hosted continuous integration services

Problems with Cloud Execution



Benchmark

Cloud instance

Unreliable results

Virtual machines or containers

Co-located neighbors

No control over hw/sw optimizations

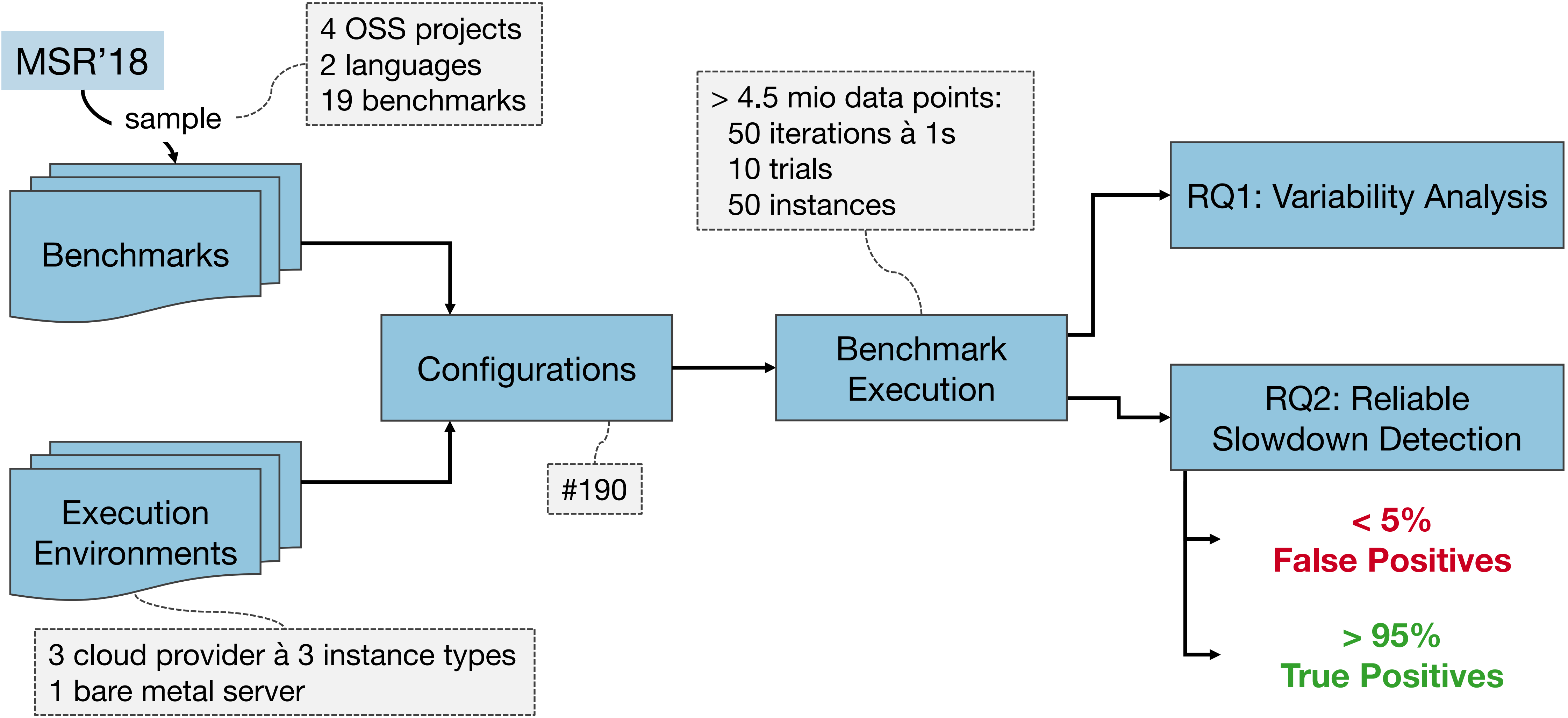
Background services (e.g., monitoring)

Empirically study microbenchmark executions in unreliable environments and **simulate** detectable slowdowns

RQ 1 *How **variable** are microbenchmarks executed in different environments?*

RQ 2 *Which **slowdown sizes** can we **reliably** detect?*

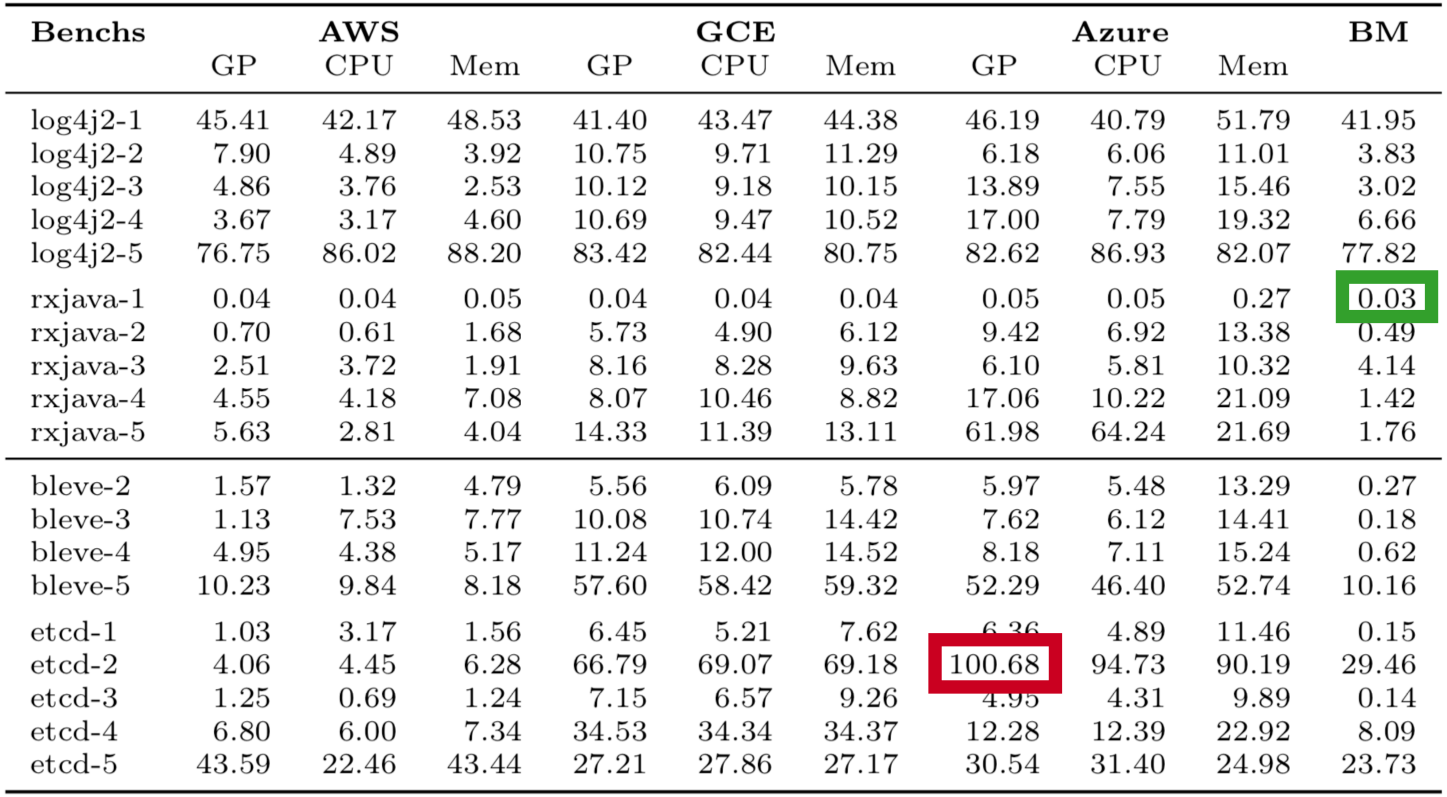
Methodology



RQ 1 *How **variable** are microbenchmarks executed in different environments?*

RQ 2 *Which **slowdown sizes** can we **reliably** detect?*

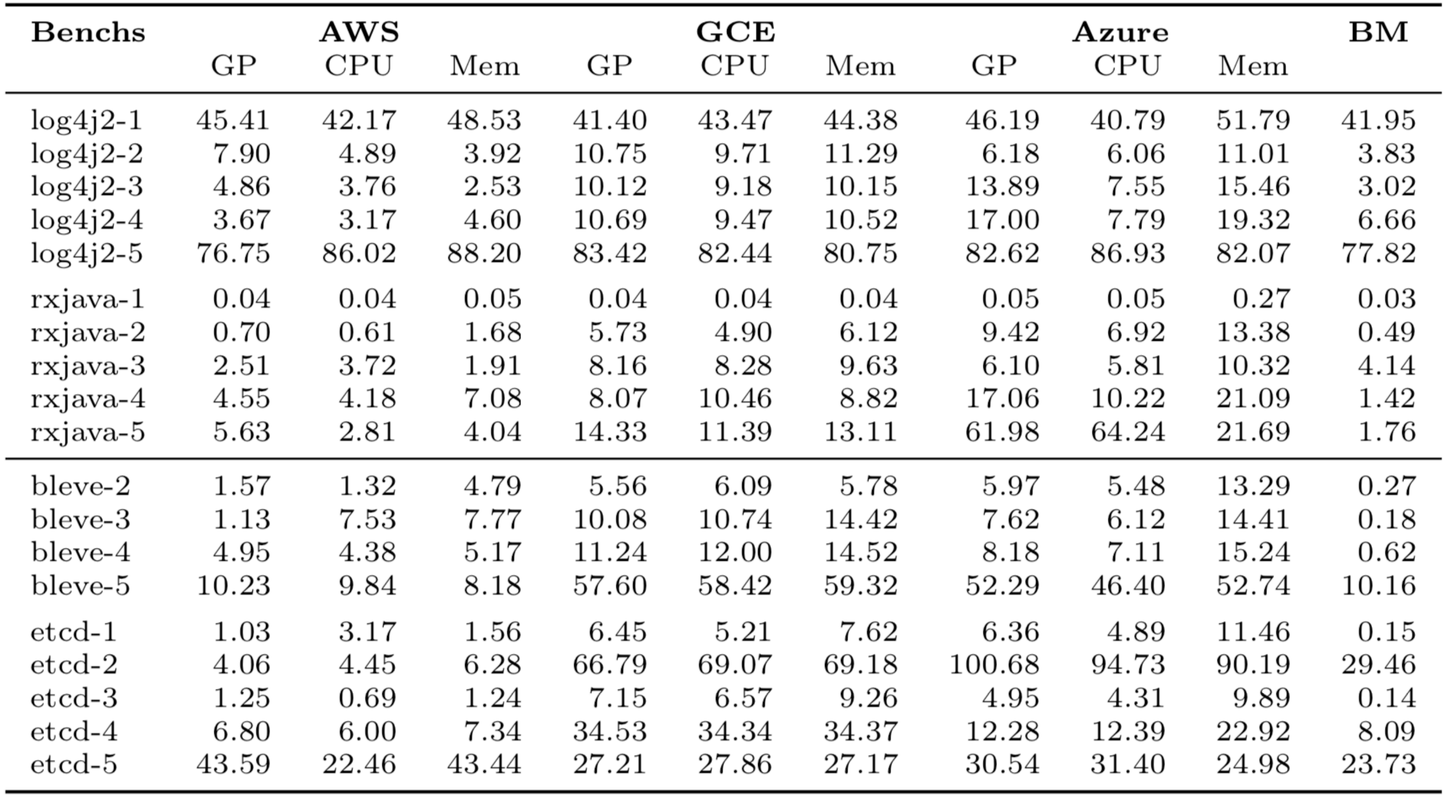
RQ 1: Variability -- Results



Benchs	AWS			GCE			Azure			BM
	GP	CPU	Mem	GP	CPU	Mem	GP	CPU	Mem	
log4j2-1	45.41	42.17	48.53	41.40	43.47	44.38	46.19	40.79	51.79	41.95
log4j2-2	7.90	4.89	3.92	10.75	9.71	11.29	6.18	6.06	11.01	3.83
log4j2-3	4.86	3.76	2.53	10.12	9.18	10.15	13.89	7.55	15.46	3.02
log4j2-4	3.67	3.17	4.60	10.69	9.47	10.52	17.00	7.79	19.32	6.66
log4j2-5	76.75	86.02	88.20	83.42	82.44	80.75	82.62	86.93	82.07	77.82
rxjava-1	0.04	0.04	0.05	0.04	0.04	0.04	0.05	0.05	0.27	0.03
rxjava-2	0.70	0.61	1.68	5.73	4.90	6.12	9.42	6.92	13.38	0.49
rxjava-3	2.51	3.72	1.91	8.16	8.28	9.63	6.10	5.81	10.32	4.14
rxjava-4	4.55	4.18	7.08	8.07	10.46	8.82	17.06	10.22	21.09	1.42
rxjava-5	5.63	2.81	4.04	14.33	11.39	13.11	61.98	64.24	21.69	1.76
bleve-2	1.57	1.32	4.79	5.56	6.09	5.78	5.97	5.48	13.29	0.27
bleve-3	1.13	7.53	7.77	10.08	10.74	14.42	7.62	6.12	14.41	0.18
bleve-4	4.95	4.38	5.17	11.24	12.00	14.52	8.18	7.11	15.24	0.62
bleve-5	10.23	9.84	8.18	57.60	58.42	59.32	52.29	46.40	52.74	10.16
etcd-1	1.03	3.17	1.56	6.45	5.21	7.62	6.36	4.89	11.46	0.15
etcd-2	4.06	4.45	6.28	66.79	69.07	69.18	100.68	94.73	90.19	29.46
etcd-3	1.25	0.69	1.24	7.15	6.57	9.26	4.95	4.31	9.89	0.14
etcd-4	6.80	6.00	7.34	34.53	34.34	34.37	12.28	12.39	22.92	8.09
etcd-5	43.59	22.46	43.44	27.21	27.86	27.17	30.54	31.40	24.98	23.73

Range between 0.03% and >100% CV

RQ 1: Variability -- Results



Benchs		AWS			GCE			Azure		BM
	GP	CPU	Mem	GP	CPU	Mem	GP	CPU	Mem	
log4j2-1	45.41	42.17	48.53	41.40	43.47	44.38	46.19	40.79	51.79	41.95
log4j2-2	7.90	4.89	3.92	10.75	9.71	11.29	6.18	6.06	11.01	3.83
log4j2-3	4.86	3.76	2.53	10.12	9.18	10.15	13.89	7.55	15.46	3.02
log4j2-4	3.67	3.17	4.60	10.69	9.47	10.52	17.00	7.79	19.32	6.66
log4j2-5	76.75	86.02	88.20	83.42	82.44	80.75	82.62	86.93	82.07	77.82
rxjava-1	0.04	0.04	0.05	0.04	0.04	0.04	0.05	0.05	0.27	0.03
rxjava-2	0.70	0.61	1.68	5.73	4.90	6.12	9.42	6.92	13.38	0.49
rxjava-3	2.51	3.72	1.91	8.16	8.28	9.63	6.10	5.81	10.32	4.14
rxjava-4	4.55	4.18	7.08	8.07	10.46	8.82	17.06	10.22	21.09	1.42
rxjava-5	5.63	2.81	4.04	14.33	11.39	13.11	61.98	64.24	21.69	1.76
bleve-2	1.57	1.32	4.79	5.56	6.09	5.78	5.97	5.48	13.29	0.27
bleve-3	1.13	7.53	7.77	10.08	10.74	14.42	7.62	6.12	14.41	0.18
bleve-4	4.95	4.38	5.17	11.24	12.00	14.52	8.18	7.11	15.24	0.62
bleve-5	10.23	9.84	8.18	57.60	58.42	59.32	52.29	46.40	52.74	10.16
etcd-1	1.03	3.17	1.56	6.45	5.21	7.62	6.36	4.89	11.46	0.15
etcd-2	4.06	4.45	6.28	66.79	69.07	69.18	100.68	94.73	90.19	29.46
etcd-3	1.25	0.69	1.24	7.15	6.57	9.26	4.95	4.31	9.89	0.14
etcd-4	6.80	6.00	7.34	34.53	34.34	34.37	12.28	12.39	22.92	8.09
etcd-5	43.59	22.46	43.44	27.21	27.86	27.17	30.54	31.40	24.98	23.73

Range between 0.03% and >100% CV

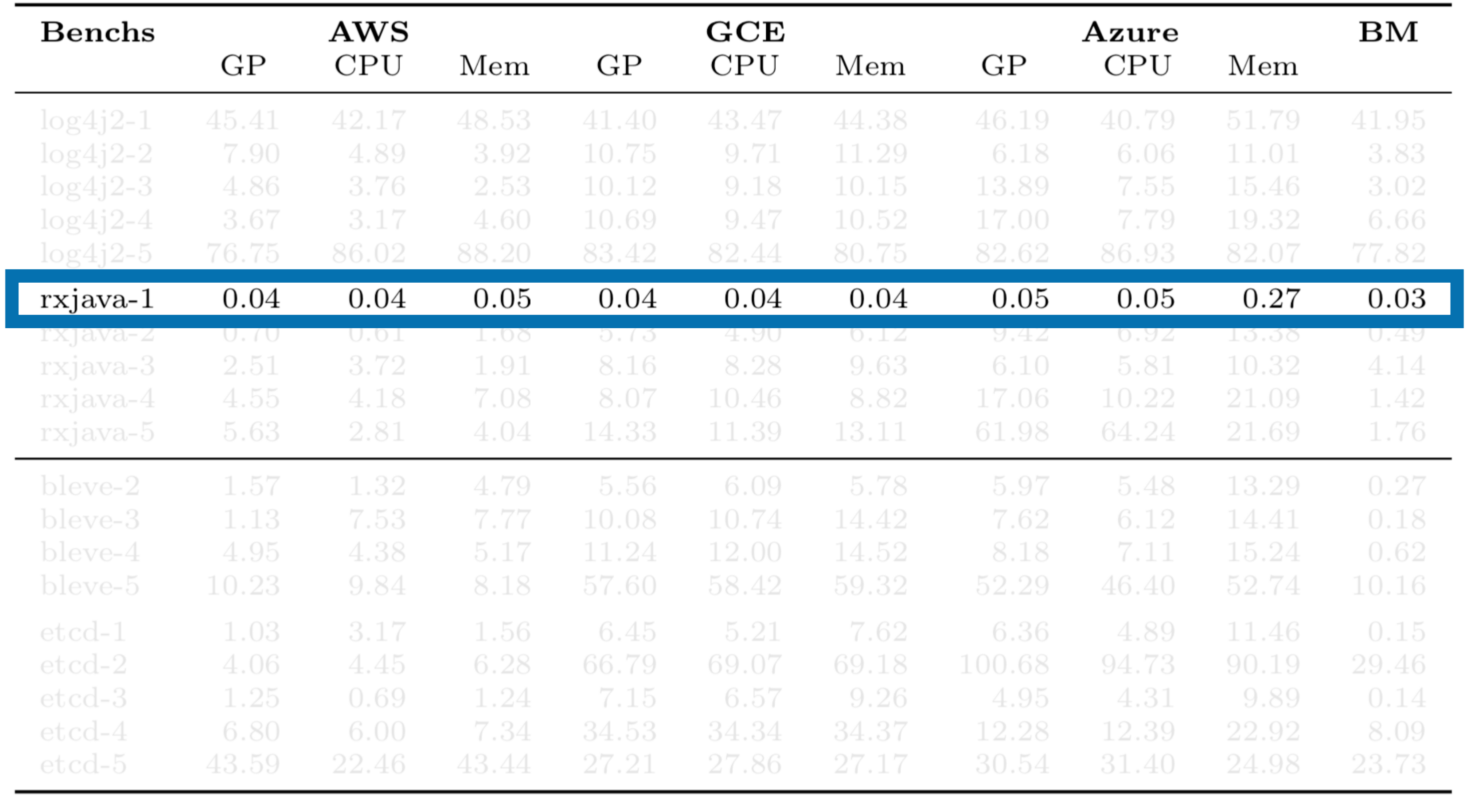
①

②

③

3 groups of benchmarks

RQ 1: Variability -- Results



Benchs	AWS			GCE			Azure			BM
	GP	CPU	Mem	GP	CPU	Mem	GP	CPU	Mem	
log4j2-1	45.41	42.17	48.53	41.40	43.47	44.38	46.19	40.79	51.79	41.95
log4j2-2	7.90	4.89	3.92	10.75	9.71	11.29	6.18	6.06	11.01	3.83
log4j2-3	4.86	3.76	2.53	10.12	9.18	10.15	13.89	7.55	15.46	3.02
log4j2-4	3.67	3.17	4.60	10.69	9.47	10.52	17.00	7.79	19.32	6.66
log4j2-5	76.75	86.02	88.20	83.42	82.44	80.75	82.62	86.93	82.07	77.82
rxjava-1	0.04	0.04	0.05	0.04	0.04	0.04	0.05	0.05	0.27	0.03
rxjava-2	0.70	0.61	1.68	5.73	4.90	6.12	9.42	6.92	13.38	0.49
rxjava-3	2.51	3.72	1.91	8.16	8.28	9.63	6.10	5.81	10.32	4.14
rxjava-4	4.55	4.18	7.08	8.07	10.46	8.82	17.06	10.22	21.09	1.42
rxjava-5	5.63	2.81	4.04	14.33	11.39	13.11	61.98	64.24	21.69	1.76
bleve-2	1.57	1.32	4.79	5.56	6.09	5.78	5.97	5.48	13.29	0.27
bleve-3	1.13	7.53	7.77	10.08	10.74	14.42	7.62	6.12	14.41	0.18
bleve-4	4.95	4.38	5.17	11.24	12.00	14.52	8.18	7.11	15.24	0.62
bleve-5	10.23	9.84	8.18	57.60	58.42	59.32	52.29	46.40	52.74	10.16
etcd-1	1.03	3.17	1.56	6.45	5.21	7.62	6.36	4.89	11.46	0.15
etcd-2	4.06	4.45	6.28	66.79	69.07	69.18	100.68	94.73	90.19	29.46
etcd-3	1.25	0.69	1.24	7.15	6.57	9.26	4.95	4.31	9.89	0.14
etcd-4	6.80	6.00	7.34	34.53	34.34	34.37	12.28	12.39	22.92	8.09
etcd-5	43.59	22.46	43.44	27.21	27.86	27.17	30.54	31.40	24.98	23.73

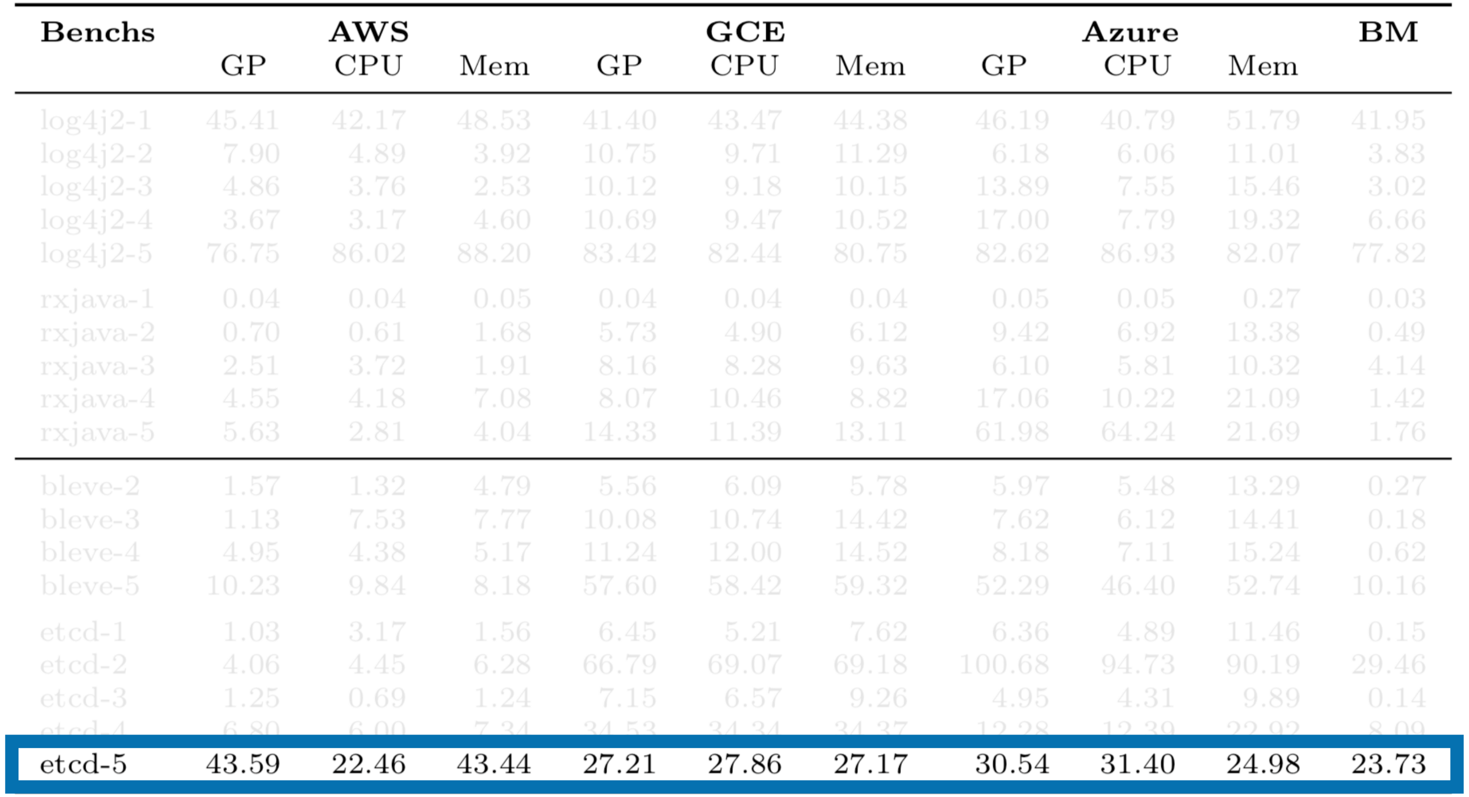
Range between 0.03% and >100% CV

1. No variability => stable

2.

3.

RQ 1: Variability -- Results



Benchs		AWS			GCE			Azure		BM
	GP	CPU	Mem	GP	CPU	Mem	GP	CPU	Mem	
log4j2-1	45.41	42.17	48.53	41.40	43.47	44.38	46.19	40.79	51.79	41.95
log4j2-2	7.90	4.89	3.92	10.75	9.71	11.29	6.18	6.06	11.01	3.83
log4j2-3	4.86	3.76	2.53	10.12	9.18	10.15	13.89	7.55	15.46	3.02
log4j2-4	3.67	3.17	4.60	10.69	9.47	10.52	17.00	7.79	19.32	6.66
log4j2-5	76.75	86.02	88.20	83.42	82.44	80.75	82.62	86.93	82.07	77.82
rxjava-1	0.04	0.04	0.05	0.04	0.04	0.04	0.05	0.05	0.27	0.03
rxjava-2	0.70	0.61	1.68	5.73	4.90	6.12	9.42	6.92	13.38	0.49
rxjava-3	2.51	3.72	1.91	8.16	8.28	9.63	6.10	5.81	10.32	4.14
rxjava-4	4.55	4.18	7.08	8.07	10.46	8.82	17.06	10.22	21.09	1.42
rxjava-5	5.63	2.81	4.04	14.33	11.39	13.11	61.98	64.24	21.69	1.76
bleve-2	1.57	1.32	4.79	5.56	6.09	5.78	5.97	5.48	13.29	0.27
bleve-3	1.13	7.53	7.77	10.08	10.74	14.42	7.62	6.12	14.41	0.18
bleve-4	4.95	4.38	5.17	11.24	12.00	14.52	8.18	7.11	15.24	0.62
bleve-5	10.23	9.84	8.18	57.60	58.42	59.32	52.29	46.40	52.74	10.16
etcd-1	1.03	3.17	1.56	6.45	5.21	7.62	6.36	4.89	11.46	0.15
etcd-2	4.06	4.45	6.28	66.79	69.07	69.18	100.68	94.73	90.19	29.46
etcd-3	1.25	0.69	1.24	7.15	6.57	9.26	4.95	4.31	9.89	0.14
etcd-4	6.80	6.00	7.34	34.53	34.34	34.37	12.28	12.39	22.92	8.09
etcd-5	43.59	22.46	43.44	27.21	27.86	27.17	30.54	31.40	24.98	23.73

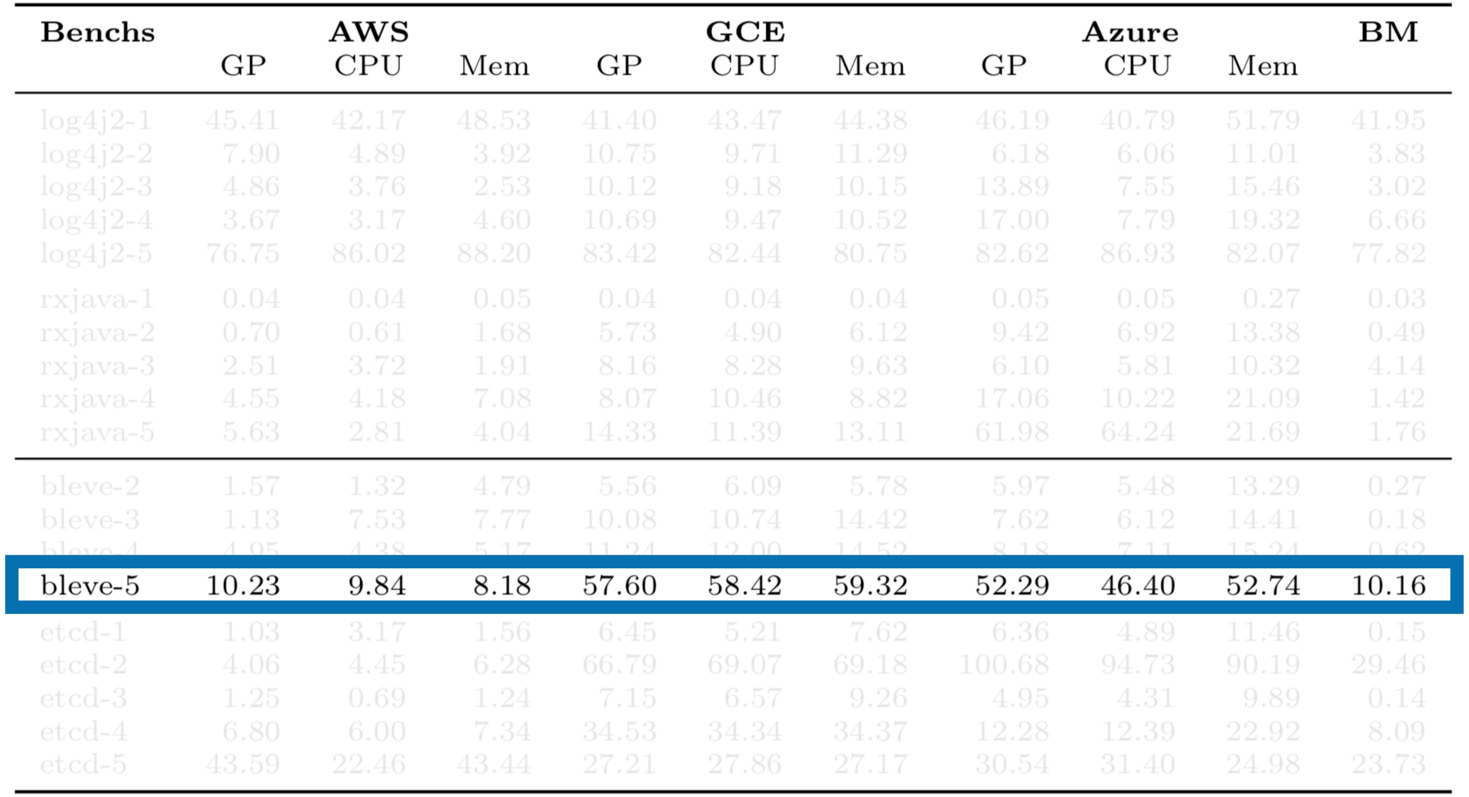
Range between 0.03% and >100% CV

1 No variability => stable

2 Variable in all environments

3

RQ 1: Variability -- Results

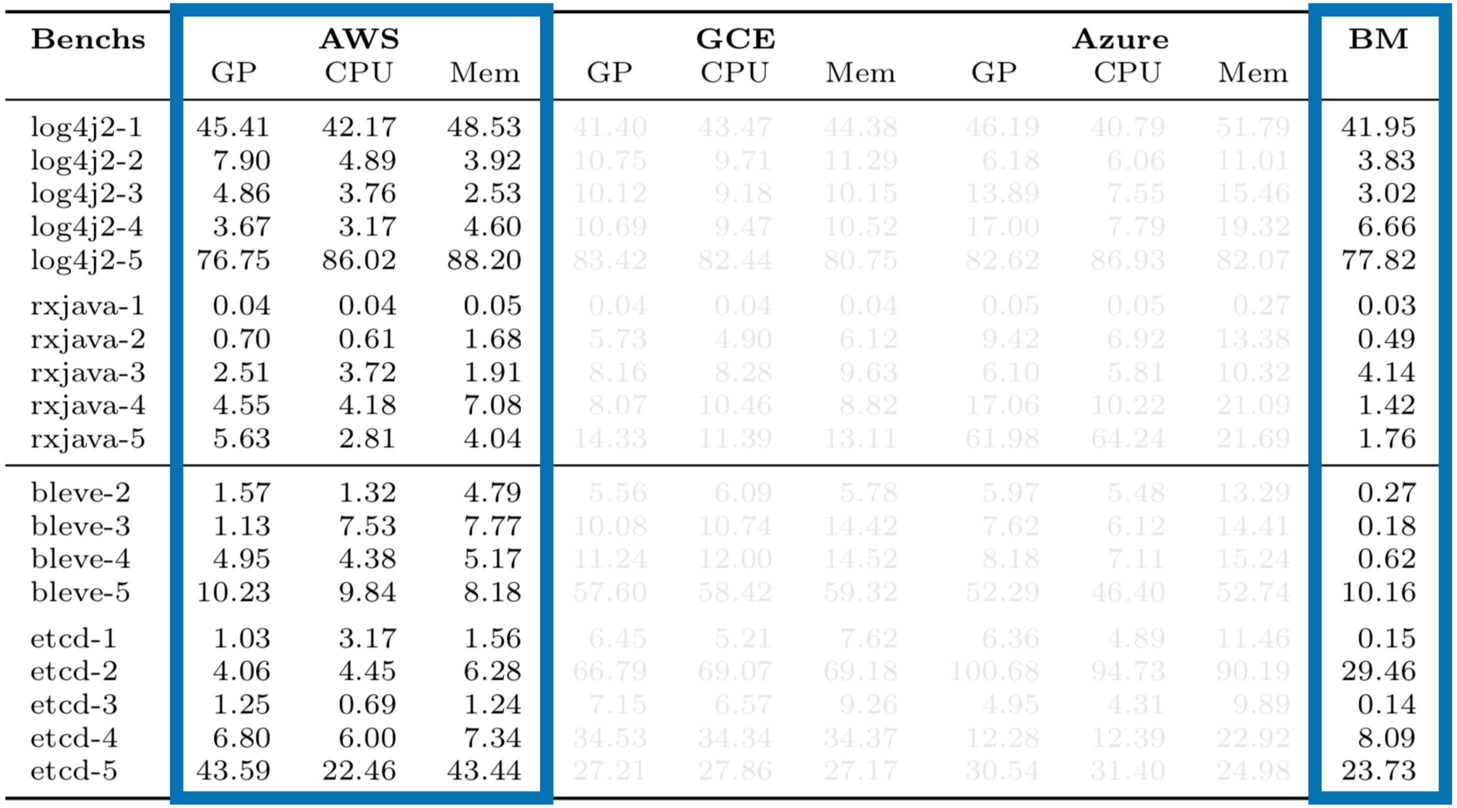


Benchs	AWS			GCE			Azure			BM
	GP	CPU	Mem	GP	CPU	Mem	GP	CPU	Mem	
log4j2-1	45.41	42.17	48.53	41.40	43.47	44.38	46.19	40.79	51.79	41.95
log4j2-2	7.90	4.89	3.92	10.75	9.71	11.29	6.18	6.06	11.01	3.83
log4j2-3	4.86	3.76	2.53	10.12	9.18	10.15	13.89	7.55	15.46	3.02
log4j2-4	3.67	3.17	4.60	10.69	9.47	10.52	17.00	7.79	19.32	6.66
log4j2-5	76.75	86.02	88.20	83.42	82.44	80.75	82.62	86.93	82.07	77.82
rxjava-1	0.04	0.04	0.05	0.04	0.04	0.04	0.05	0.05	0.27	0.03
rxjava-2	0.70	0.61	1.68	5.73	4.90	6.12	9.42	6.92	13.38	0.49
rxjava-3	2.51	3.72	1.91	8.16	8.28	9.63	6.10	5.81	10.32	4.14
rxjava-4	4.55	4.18	7.08	8.07	10.46	8.82	17.06	10.22	21.09	1.42
rxjava-5	5.63	2.81	4.04	14.33	11.39	13.11	61.98	64.24	21.69	1.76
bleve-2	1.57	1.32	4.79	5.56	6.09	5.78	5.97	5.48	13.29	0.27
bleve-3	1.13	7.53	7.77	10.08	10.74	14.42	7.62	6.12	14.41	0.18
bleve-4	4.95	4.38	5.17	11.24	12.00	14.52	8.18	7.11	15.24	0.62
bleve-5	10.23	9.84	8.18	57.60	58.42	59.32	52.29	46.40	52.74	10.16
etcd-1	1.03	3.17	1.56	6.45	5.21	7.62	6.36	4.89	11.46	0.15
etcd-2	4.06	4.45	6.28	66.79	69.07	69.18	100.68	94.73	90.19	29.46
etcd-3	1.25	0.69	1.24	7.15	6.57	9.26	4.95	4.31	9.89	0.14
etcd-4	6.80	6.00	7.34	34.53	34.34	34.37	12.28	12.39	22.92	8.09
etcd-5	43.59	22.46	43.44	27.21	27.86	27.17	30.54	31.40	24.98	23.73

Range between 0.03% and >100% CV

1. No variability => stable
2. Variable in all environments
3. Variability changes

RQ 1: Variability -- Results



Benchs	AWS			GCE			Azure			BM
	GP	CPU	Mem	GP	CPU	Mem	GP	CPU	Mem	
log4j2-1	45.41	42.17	48.53	41.40	43.47	44.38	46.19	40.79	51.79	41.95
log4j2-2	7.90	4.89	3.92	10.75	9.71	11.29	6.18	6.06	11.01	3.83
log4j2-3	4.86	3.76	2.53	10.12	9.18	10.15	13.89	7.55	15.46	3.02
log4j2-4	3.67	3.17	4.60	10.69	9.47	10.52	17.00	7.79	19.32	6.66
log4j2-5	76.75	86.02	88.20	83.42	82.44	80.75	82.62	86.93	82.07	77.82
rxjava-1	0.04	0.04	0.05	0.04	0.04	0.04	0.05	0.05	0.27	0.03
rxjava-2	0.70	0.61	1.68	5.73	4.90	6.12	9.42	6.92	13.38	0.49
rxjava-3	2.51	3.72	1.91	8.16	8.28	9.63	6.10	5.81	10.32	4.14
rxjava-4	4.55	4.18	7.08	8.07	10.46	8.82	17.06	10.22	21.09	1.42
rxjava-5	5.63	2.81	4.04	14.33	11.39	13.11	61.98	64.24	21.69	1.76
bleve-2	1.57	1.32	4.79	5.56	6.09	5.78	5.97	5.48	13.29	0.27
bleve-3	1.13	7.53	7.77	10.08	10.74	14.42	7.62	6.12	14.41	0.18
bleve-4	4.95	4.38	5.17	11.24	12.00	14.52	8.18	7.11	15.24	0.62
bleve-5	10.23	9.84	8.18	57.60	58.42	59.32	52.29	46.40	52.74	10.16
etcd-1	1.03	3.17	1.56	6.45	5.21	7.62	6.36	4.89	11.46	0.15
etcd-2	4.06	4.45	6.28	66.79	69.07	69.18	100.68	94.73	90.19	29.46
etcd-3	1.25	0.69	1.24	7.15	6.57	9.26	4.95	4.31	9.89	0.14
etcd-4	6.80	6.00	7.34	34.53	34.34	34.37	12.28	12.39	22.92	8.09
etcd-5	43.59	22.46	43.44	27.21	27.86	27.17	30.54	31.40	24.98	23.73

Range between 0.03% and >100% CV

1. No variability => stable
2. Variable in all environments
3. Variability changes

AWS and BM similarly stable

RQ 1 *How **variable** are microbenchmarks executed in different environments?*

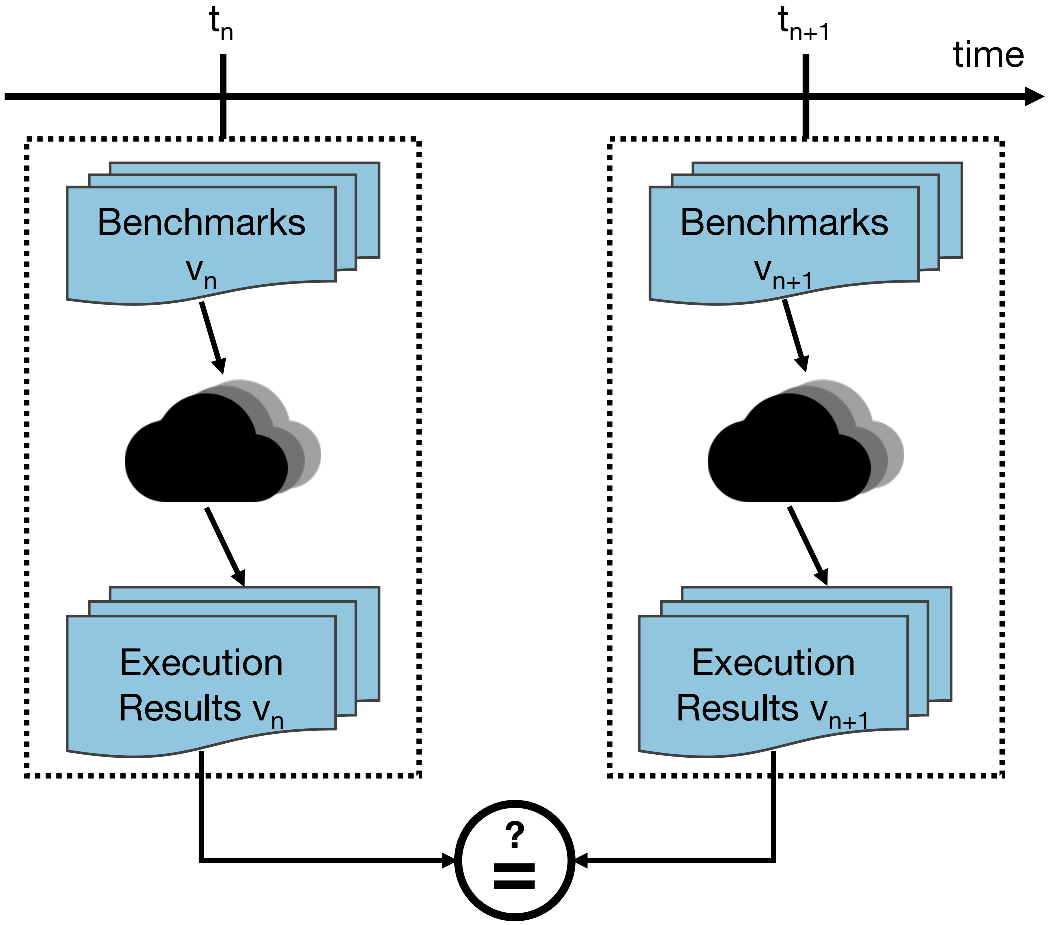
RQ 2 *Which **slowdown sizes** can we **reliably** detect?*

RQ 1 *How **variable** are microbenchmarks executed in different environments?*

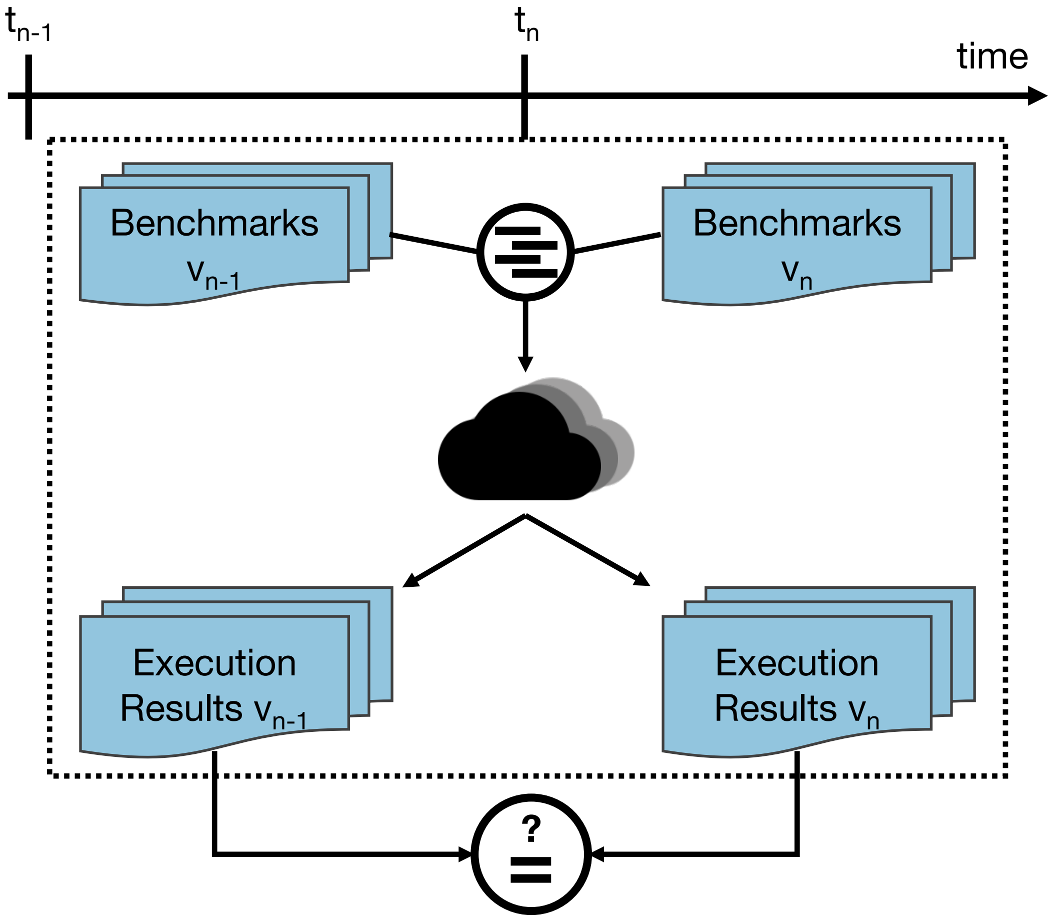
RQ 2 *Which **slowdown sizes** can we **reliably** detect?*

RQ 2: Detection Simulation -- Method

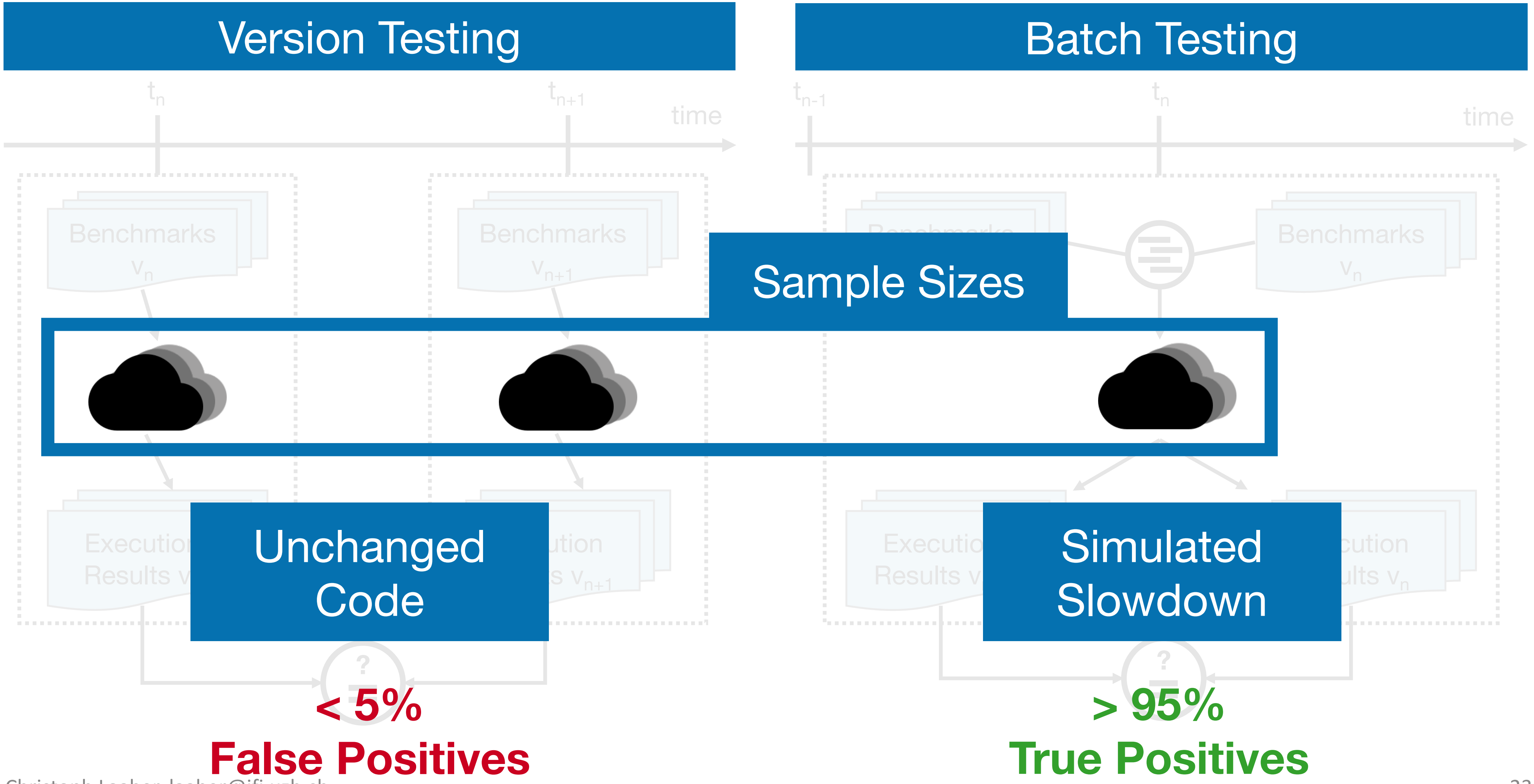
Version Testing



Batch Testing



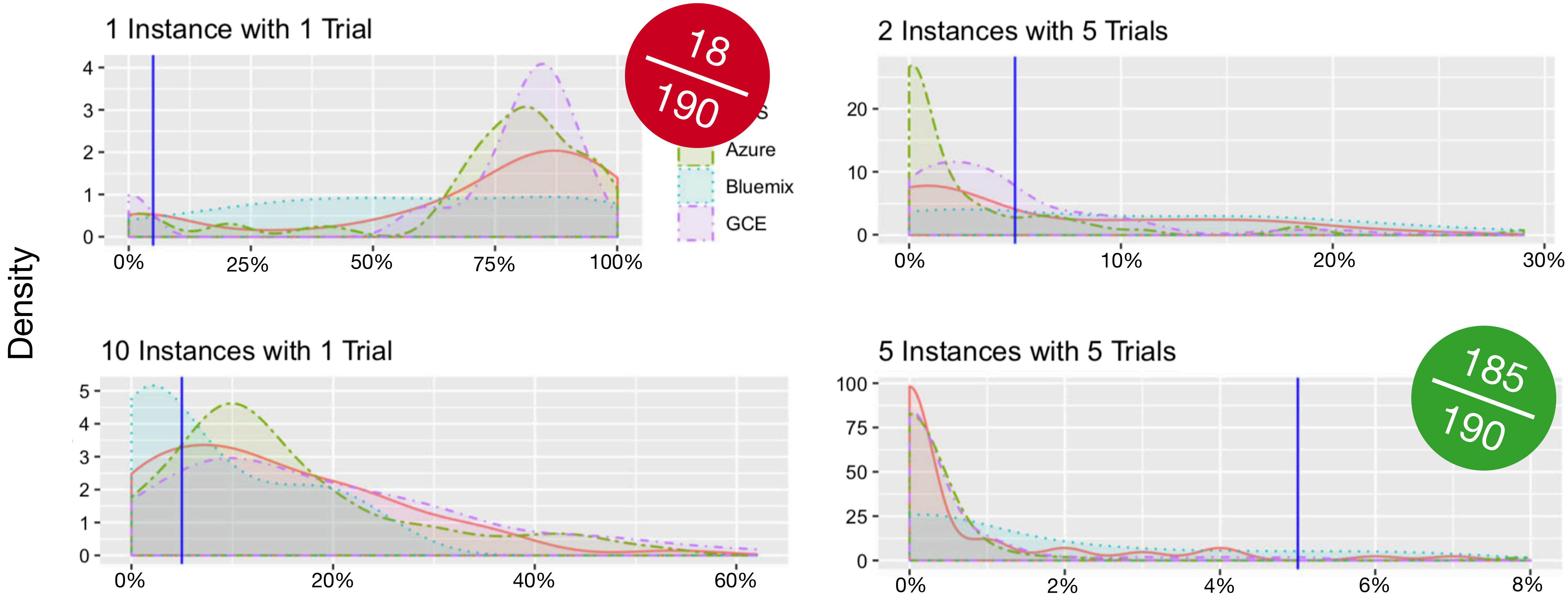
RQ 2: Detection Simulation -- Method



RQ 2: False Positives -- Results

Version Testing

Batch Testing



1 Instance with 1 Trial

18/190

2 Instances with 5 Trials

10 Instances with 1 Trial

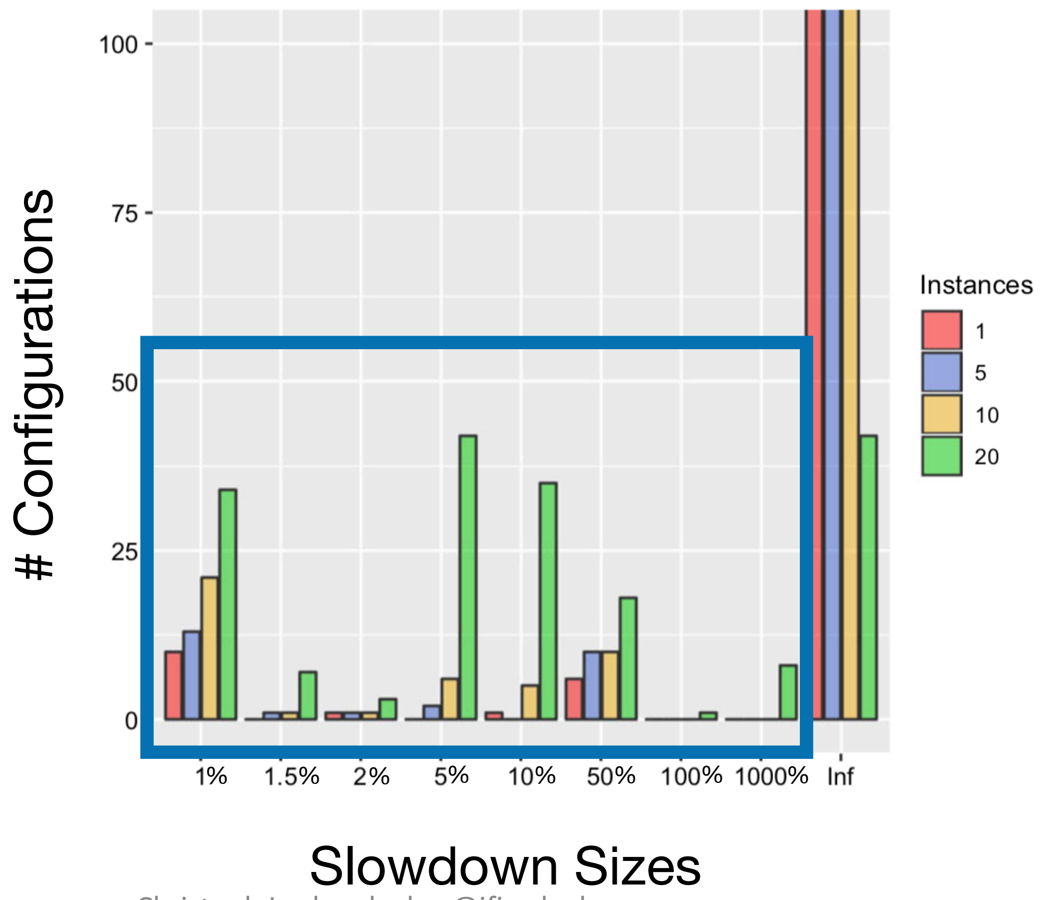
5 Instances with 5 Trials

185/190

False Positives

RQ 2: Smallest Slowdowns -- Results

Version Testing

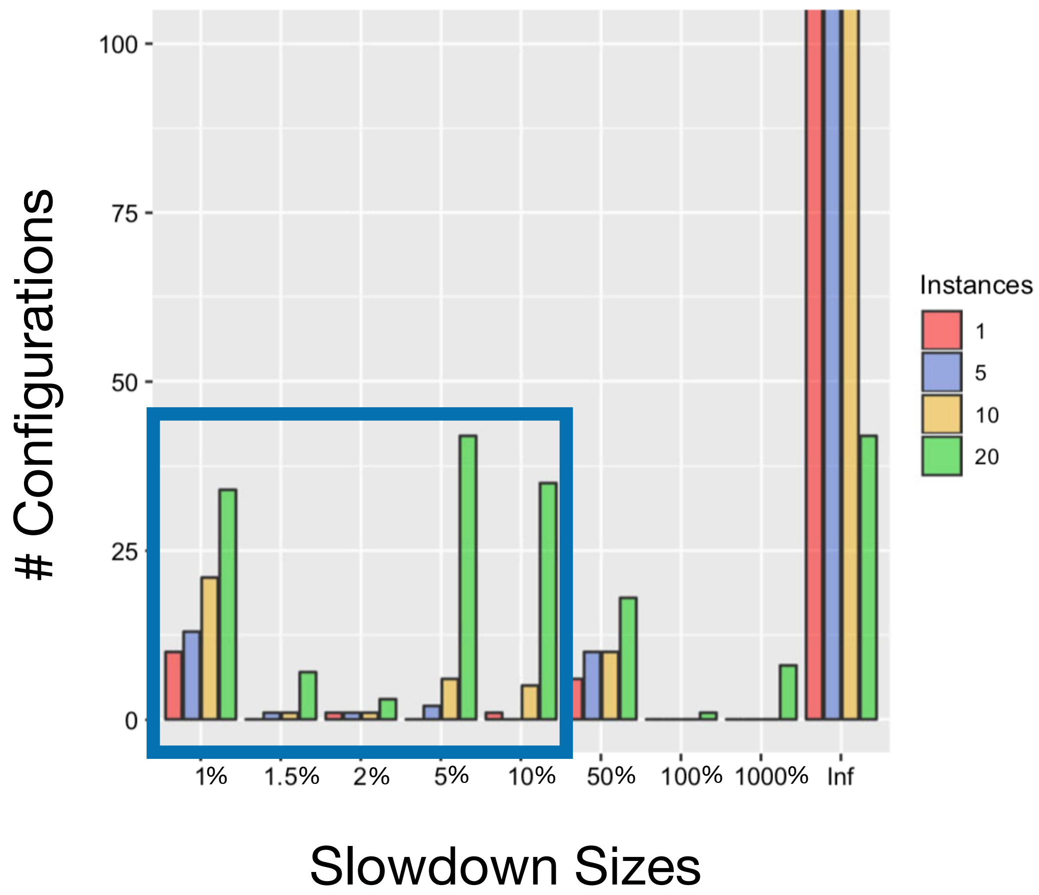


Reliable slowdown detection:

1	9%	5	15%
10	23%	20	88%

RQ 2: Smallest Slowdowns -- Results

Version Testing



Reliable slowdown detection:

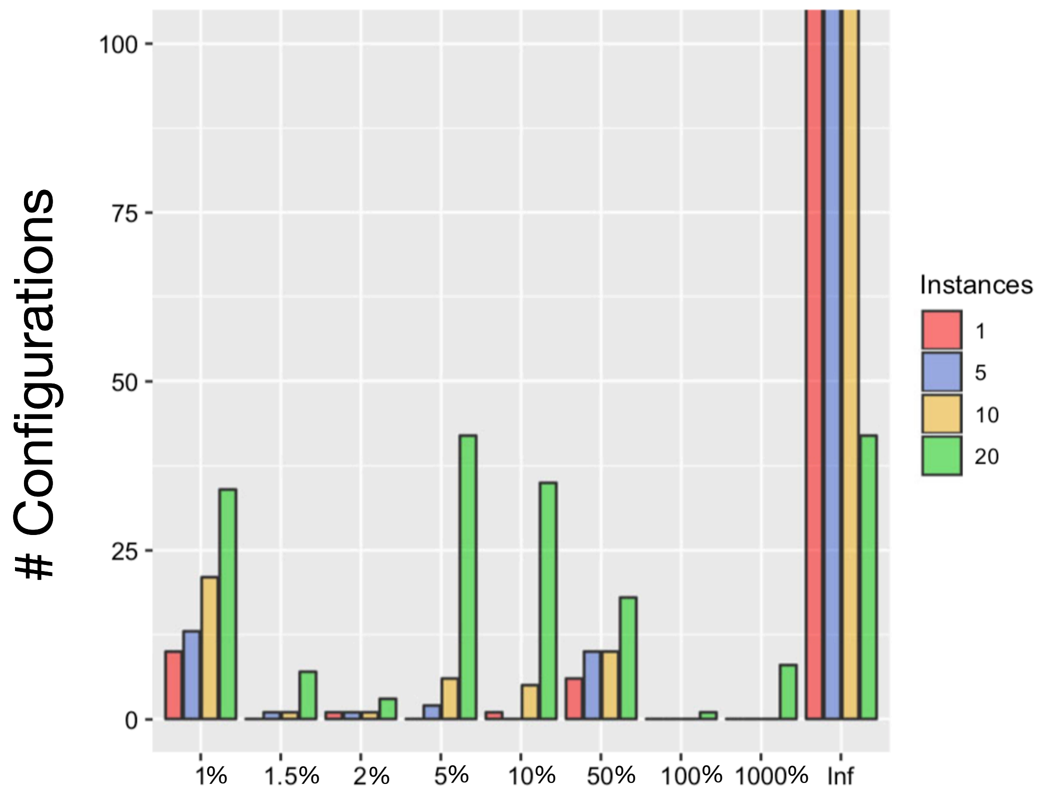
1	9%	5	15%
10	23%	20	88%

Slowdowns <= 10%:

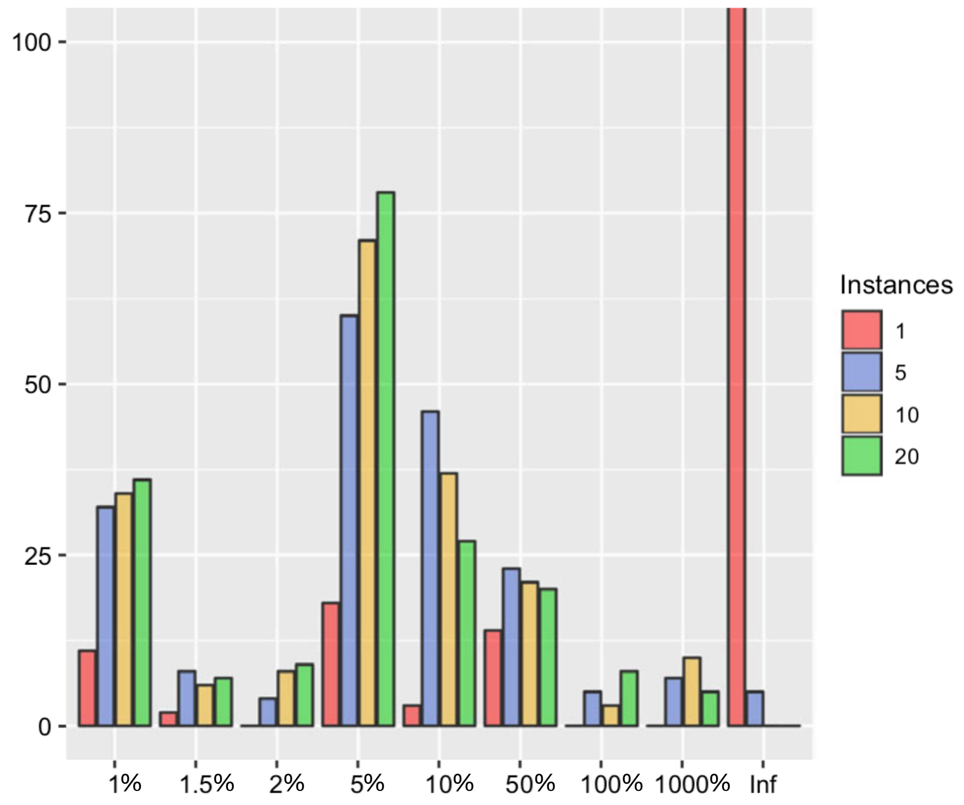
20	64% configurations

RQ 2: Smallest Slowdowns -- Results

Version Testing



Batch Testing



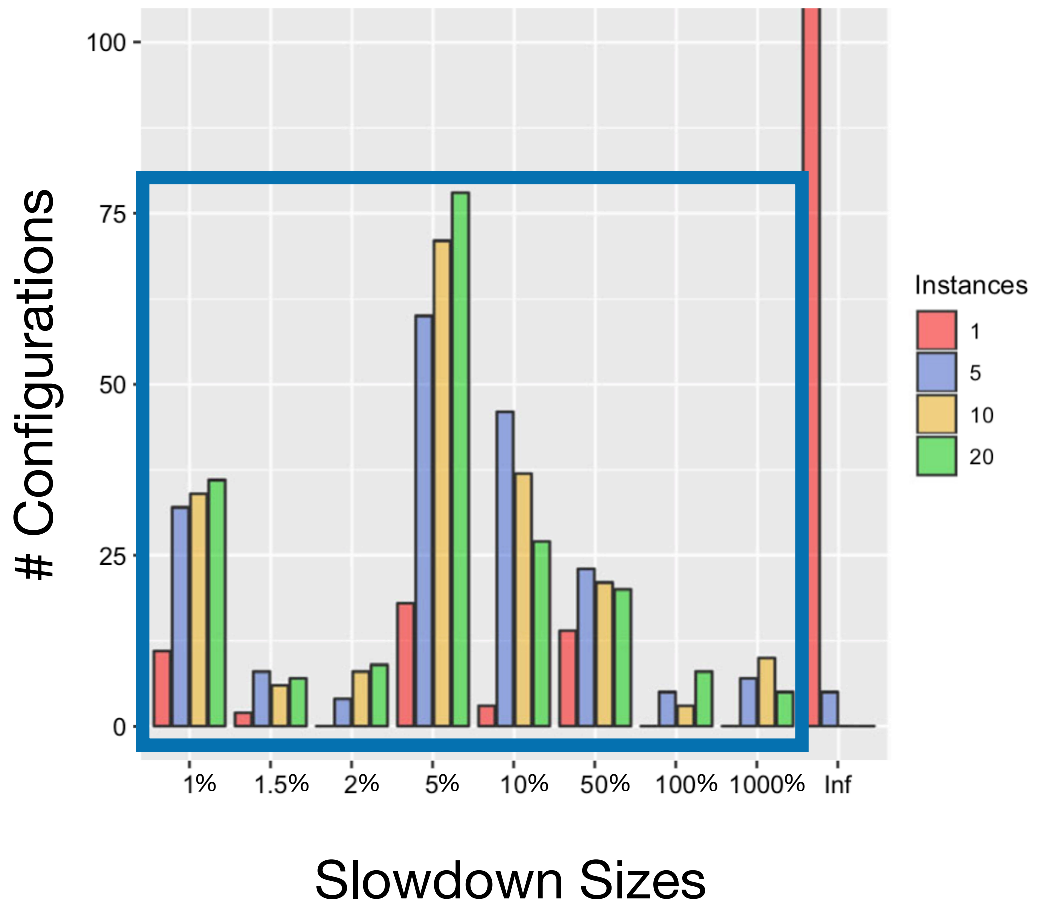
Slowdown Sizes

RQ 2: Smallest Slowdowns -- Results

Batch Testing

Reliable slowdown detection:

1	25%	5	97%
10	100%	20	100%



RQ 2: Smallest Slowdowns -- Results

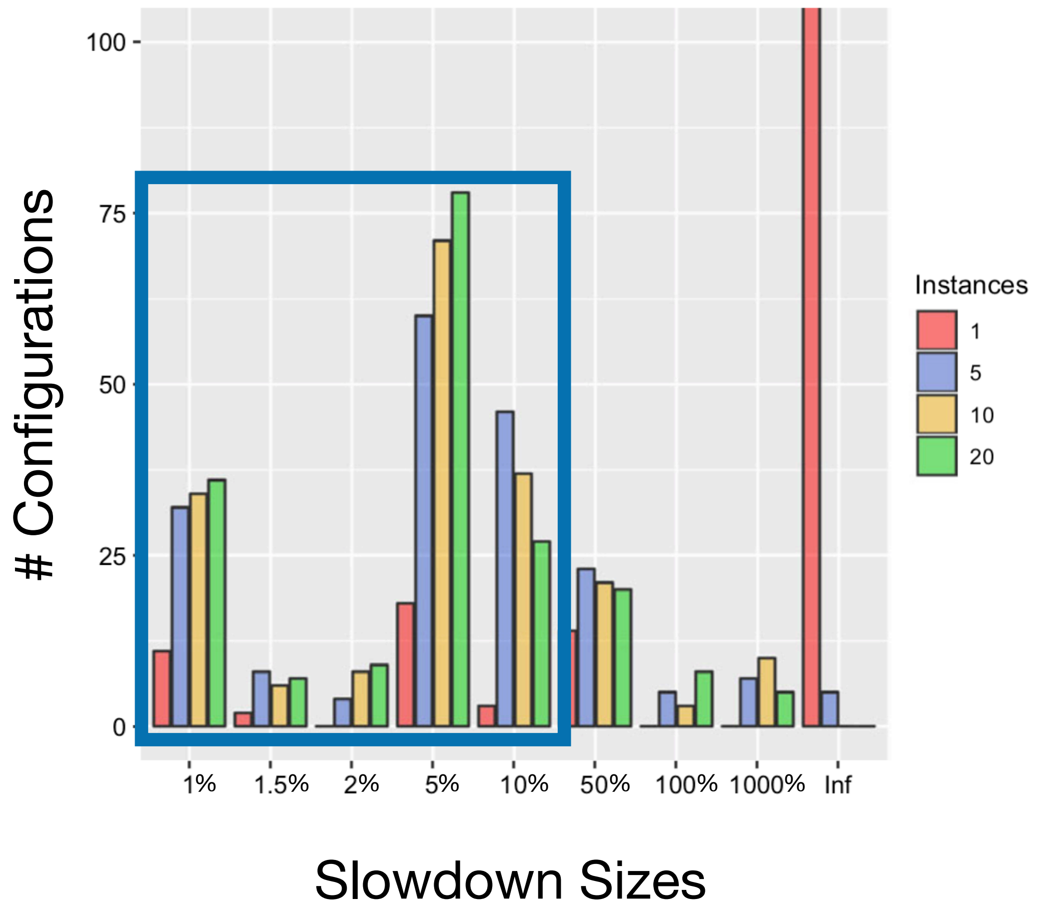
Reliable slowdown detection:

1	25%	5	97%
10	100%	20	100%

Slowdowns <= 10%:

5	79% configurations

Batch Testing



What have we learned?

IBM bare-metal and AWS instances deliver stable results

Always check for false positives

Batch testing increases reliability

Detection of 5%-10% slowdowns often possible

Future Ahead!

Help developers writing tests that have stable results

Automatically decide how often to replicate executions

Prioritize/select reliable benchmarks

Generate reliable benchmarks

Software Microbenchmarking in the Cloud. How Bad is it Really?

Christoph Laaber, Joel Scheuner, Philipp Leitner

University of Zurich UZH

s.e.a.l. software evolution & architecture lab

ICET LAB

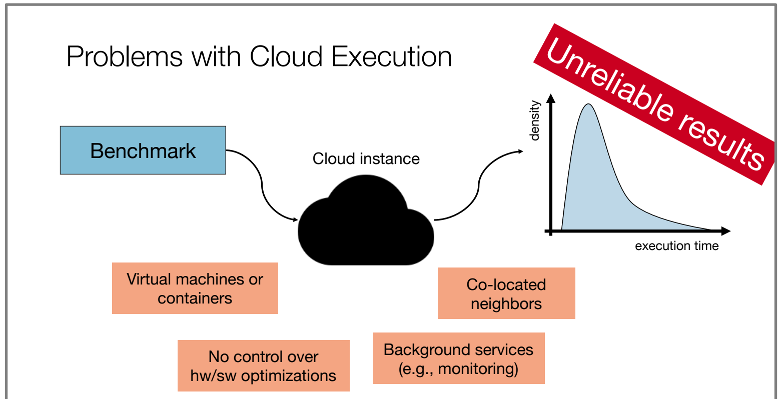
CHALMERS

UNIVERSITY OF GOTHENBURG

Why Execute Benchmarks in the Cloud then?

- Unavailability of / no training for bare-metal machines
- Long benchmarking run times
- Little set-up and maintenance effort
- Hosted continuous integration services

Problems with Cloud Execution



Unreliable results

Benchmark

Cloud instance

Virtual machines or containers

Co-located neighbors

No control over hw/sw optimizations

Background services (e.g., monitoring)

RQ 1: Variability -- Results

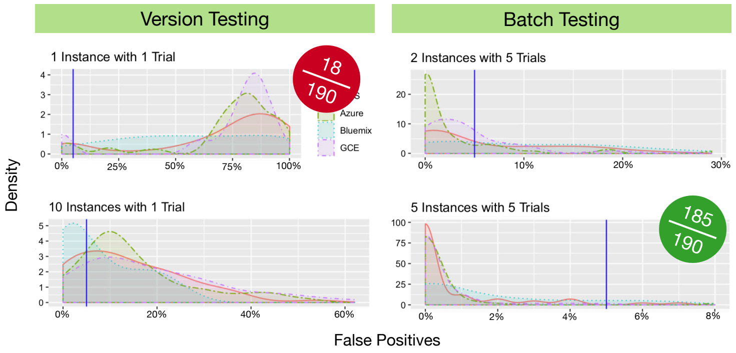
Benchs	AWS			GCE			Azure			BM
	GP	CPU	Mem	GP	CPU	Mem	GP	CPU	Mem	
log4j2-1	45.41	42.17	48.53	41.40	43.47	44.38	46.19	40.79	51.79	41.95
log4j2-2	7.90	4.89	3.92	10.75	9.71	11.29	6.18	6.06	11.01	3.83
log4j2-3	4.86	3.76	2.53	10.12	9.18	10.15	13.89	7.55	15.46	3.02
log4j2-4	3.67	3.17	4.60	10.69	9.47	10.52	17.00	7.79	19.32	6.66
log4j2-5	76.75	86.02	88.20	83.42	82.44	80.75	82.62	86.93	82.07	77.82
rxjava-1	0.04	0.04	0.05	0.04	0.04	0.04	0.05	0.05	0.27	0.03
rxjava-2	0.70	0.61	1.68	5.73	4.90	6.12	9.42	6.92	13.38	0.49
rxjava-3	2.51	3.72	1.91	8.16	8.28	9.63	6.10	5.81	10.32	4.14
rxjava-4	4.55	4.18	7.08	8.07	10.46	8.82	17.06	10.22	21.09	1.42
rxjava-5	5.63	2.81	4.04	14.33	11.39	13.11	61.98	64.24	21.69	1.76
bleve-2	1.57	1.32	4.79	5.56	6.09	5.78	5.97	5.48	13.29	0.27
bleve-3	1.13	7.53	7.77	10.08	10.74	14.42	7.62	6.12	14.41	0.18
bleve-4	4.95	4.38	5.17	11.24	12.00	14.52	8.18	7.11	15.24	0.62
bleve-5	10.23	9.84	8.18	57.60	58.42	59.32	52.29	46.40	52.74	10.16
etcd-1	1.03	3.17	1.56	6.45	5.21	7.62	6.36	4.89	11.46	0.15
etcd-2	4.06	4.45	6.28	66.79	69.07	69.18	100.68	94.73	90.19	29.46
etcd-3	1.25	0.69	1.24	7.15	6.57	9.26	4.95	4.31	9.89	0.14
etcd-4	6.80	6.00	7.34	34.53	34.34	34.37	12.28	12.39	22.92	8.09
etcd-5	43.59	22.46	43.44	27.21	27.86	27.17	30.54	31.40	24.98	23.73

Range between 0.03% and >100% CV

1. No variability => stable
2. Variable in all environments
3. Variability changes

AWS and BM similarly stable

RQ 2: False Positives -- Results



Version Testing

Batch Testing

1 Instance with 1 Trial

2 Instances with 5 Trials

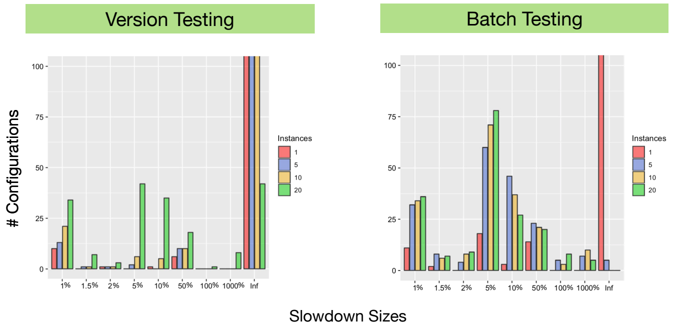
10 Instances with 1 Trial

5 Instances with 5 Trials

18/190

185/190

RQ 2: Smallest Slowdowns -- Results



Version Testing

Batch Testing

Future Ahead!

- Help developers write tests that have stable results
- Automatically decide how often to replicate executions
- Prioritize/select benchmarks that are reliable
- Generate benchmarks that are reliable

laaber@ifi.uzh.ch @ChristophLaaber

https://doi.org/10.1007/s10664-019-09681-1 http://t.uzh.ch/T4

Paper, Scripts, and Data

Empirical Software Engineering (2019) 24:2469–2508
https://doi.org/10.1007/s10664-019-09681-1

Check for updates

Software microbenchmarking in the cloud. How bad is it really?

Christoph Laaber[1] · Joel Scheuner[2] · Philipp Leitner[2]

Published online: 17 April 2019

Abstract
Rigorous performance engineering traditionally assumes measuring on bare-metal environments to control for as many confounding factors as possible. Unfortunately, some researchers and practitioners might not have access, knowledge, or funds to operate dedicated performance-testing hardware, making public clouds an attractive alternative. However, shared public cloud environments are inherently unpredictable in terms of the system performance they provide. In this study, we explore the effects of cloud environments on the variability of performance test results and to what extent slowdowns can still be reliably detected even in a public cloud. We focus on software microbenchmarks as an example of performance tests and execute extensive experiments on three different well-known public cloud services (AWS, GCE, and Azure) using three different cloud instance types per service. We also compare the results to a hosted bare-metal offering from IBM Bluemix. In total, we gathered more than 4.5 million unique microbenchmarking data points from benchmarks written in Java and Go. We find that the variability of results differs substantially between benchmarks and instance types (by a coefficient of variation from 0.03% to >100%). However, executing test and control experiments on the same instances (in randomized order) allows us to detect slowdowns of 10% or less with high confidence, using state-of-the-art statistical tests (i.e., Wilcoxon rank-sum and overlapping bootstrapped confidence intervals). Finally, our results indicate that Wilcoxon rank-sum manages to detect smaller slowdowns in cloud environments.

Keywords Performance testing · Microbenchmarking · Cloud · Performance-regression detection

Communicated by: Vittorio Cortellessa

✉ Christoph Laaber
laaber@ifi.uzh.ch

Joel Scheuner
scheuner@chalmers.se

Philipp Leitner
philipp.leitner@chalmers.se

[1] Department of Informatics, University of Zurich, Zurich, Switzerland
[2] Software Engineering Division, Chalmers | University of Gothenburg, Gothenburg, Sweden

Springer

Paper:

https://doi.org/10.1007/s10664-019-09681-1

Preprint:

http://t.uzh.ch/T4

Replication package:

https://doi.org/10.6084/m9.figshare.7546703